Linux Operating System

Free & Open source

- Freeware:
  - Is a software provided free of charge, but in executable form only.
- Free Software:
  - Permitting anyone to copy and modify a work, though under certain strict terms and conditions.
- Open source Software:
  - A term was introduced to be a marketing term for “free software”.
  - "free software" has always refered to *freedom* to share, copy or improve , not price.

Linux

- By computer science student Linus Torvalds in 1990.
- Minix is a small Unix-like operating system for personal computers
- Calling it "Linux" is a play on both Minix and his own name.

Anti-Unix approach

- Most operating systems are designed with a concept of files.
- Come with a set of utility programs for handling these files, and then leave it to the large *applications* to do the interesting work.
- Ex: a word processor, a spreadsheet, a presentation designer, a Web browser.

Anti-Unix approach

Most applications have:

- "open file" command to read a file from disk and open it in the application.
- commands for searching and replacing text.
- checking spelling.
- printing the current document.

- The program source code for handling all of these tasks must be accounted for separately, inside each application -- taking up extra space both in memory and on disk.

Anti-Unix approach

- the code for the same function inside of different applications must be developed from scratch, separately and independently of the others each time.
- so the progress of society as a whole is set back by the countless man-hours of time and energy programmers must waste by inefficiently reinventing all the same software functions to perform the same tasks, over and over again.
- So the progress of society as a whole is set back by hours of time and energy.
- Waste by reinventing all the same software functions to perform the same tasks, over and over again.

Unix-like operating systems

- they come with many small programs called *tools*.
- Each tool is generally capable of performing a very simple, specific task, and performing it well.
- One tool does nothing but output the file(s) or data passed to it.
- One tool spools its input to the print queue, one tool sorts the lines of its input, and so on.

Unix-like operating systems

- "pipes" a way to pass the output of one tool to the input of another.
- By knowing what the individual tools do and how they are combined, a user could now build powerful "strings" of commands.

Synergy

- Just as the tensile strength of steel is greater than the added strength of its components -- nickel, cadmium, and iron.
- Multiple tools could be combined to perform a task unpredicted by the function of the individual tools.
- Ex:
  - list the number of users currently on the system

```
who | wc -l
```

Tools

- Some tasks require more than one tool or command sequence.
- So we still haven't avoided applications entirely; Linux-based systems still have them, from editors to browsers.
- But our applications use open file formats, and we can use all of our tools on these data files.
- At the time of this writing, there were a total of 1,190 tools in the two primary tool directories (`/bin' and `/usr/bin') on the Linux system.

Examples of commands

»

calendar

- Show this month calendar
  - `$ cal`
- Show previous, current& next month calendar
  - `$ cal -3`
- Show calendar for the whole year
  - `$ cal -y`

Show file contents

- Show file contents
  - **$ cat /etc/passwd**
- Show file contents page by page
  - **$ cat /etc/passwd | more**
  - **For exit : ctrl +c**
- Show the first 10 lines of the file
  - **$ head /etc/passwd**
- Show the first n lines of the file
  - **head –n /etc/passwd**
- Show the last 10 lines of the file
  - **$ tail /etc/passwd**
- Show the last n lines of the file
  - **$ tail –n /etc/passwd**
- Show lines from 10 to 15
  - **head –15 /etc/passwd |tail -5**

Count lines, words and letters

- Count lines, words and letters in a file
  - **$ wc /etc/passwd**
- Count lines
  - **$ wc -l /etc/passwd**
- Count words
  - **$ wc -w /etc/passwd**
- Count letters
  - **$ wc -c /etc/passwd**
- **Pwd(Power Working Directory)**
  - **$ pwd**
    /home/user
- **Cd :change directory**

What every linux user knows

Accounts & Privileges

- Linux is a multi-user system, meaning that many users can use one Linux system simultaneously, from different terminals.
- So to avoid confusion (and to maintain a semblance of privacy), each user's workspace must be kept separate from the others.
- This separation is accomplished by giving each individual user an *account* on the system
- It is the name that the system (and those who use it) will then forever know you as; it's a single word, in all lowercase letters.

Accounts & Privileges

- The system administrator has a special account whose username is root.
- This account has total access to the entire system, so it is often called the *superuser*.
- A case sensitive password is also needed.

Changing your password

passwd [RET]

Changing password for kurt

Old password: ***your current password [RET]***

New password: ***your new password [RET]***

Re-enter new password: ***your new password [RET]***

Password changed.

Listing User Activity

- Finding out who you are:
  - **$ whoami [RET]**
  - **$ sudo who am i**
  - **user pts/0 2014-12-07 15:39 (:0.0)**
- Listing who is on the system.
  - **$ who [RET]**
- Listing when a user last logged on.
  - **$ last [RET]**
  - **$ last kurt [RET]**

Processes

- To list the processes in your current shell session
  - **$ ps [RET]**
- To list all the processes that user hst has running on the system:
  - **$ ps -u hst [RET]**
- To list all of the processes and give their usernames:
  - **$ ps aux [RET]**
- To display a continually updated display of the current system processes:
  - **$ top [RET]**

Processes

- To list all the processes whose commands contain reference to an `sbin' directory in them:
  - **`$ ps aux | grep sbin [RET]`**
- To list any processes whose process IDs contain a 13 in them:
  - **`$ ps aux | grep 13 [RET]`**
- To list the process whose PID is 344:
  - **`$ ps -p 344 [RET]`**

Example of Help Facilities

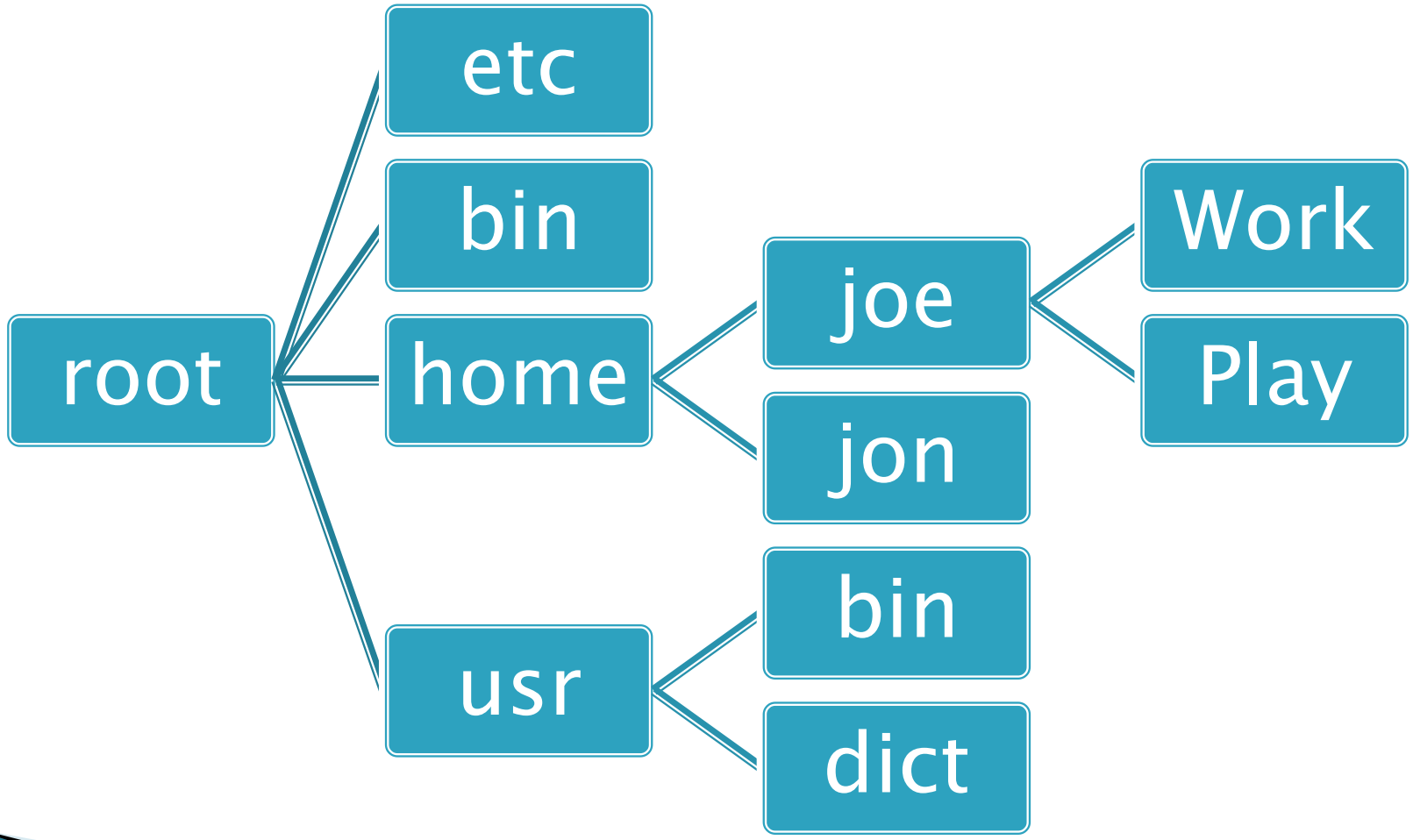
- To get a description from the manual page of the who tool:
  - **$ whatis who [RET]**
- `--help', that outputs usage information about the tool, including the options and arguments
  - **$ whoami --help [RET]**
- to view a page in the system manual
  - **$ man w [RET]**
  - **$ man -k logged**
  - **For exit from man using q.**

Files and Directories

Files and Drectories

- A *file* is a collection of data that is stored on disk and that can be manipulated as a single unit by its name.
- A *directory* is a file that acts as a folder for other files.
- A directory(parent directory) can also contain other directories (*subdirectories*).
- A *directory tree* includes a directory and all of its files, including the contents of all subdirectories.
- A slash character alone (` /') is the name of the *root directory* at the base of the directory tree hierarchy; it is the trunk from which all other files or directories branch.

directory tree



Absolute and Relative path

- if `/home/joe' is the current working directory
- `/home/joe/work/*schedule*'
- and '*work/schedule*'
- to specify `schedule'.

- `..' refers to the parent of the current working directory.
- and `.' refers to the current working directory itself.

Directory	Description
/	The ancestor of all directories on the system; all other directories are subdirectories of this directory, either directly or through other subdirectories.
/bin	Essential tools and other programs (or *binaries*).
/dev	Files representing the system's various hardware *devices*. For example, you use the file `/dev/cdrom' to access the CD-ROM drive.
/etc	Miscellaneous system configuration files, startup files, *etcetera*.
/home	The *home* directories for all of the system's users.
/lib	Essential system *library* files used by tools in `/bin'.
/proc	Files that give information about current system *processes*.
/root	The superuser's home directory, whose username is root. (In the past, the home directory for the superuser was simply `/'; later, `/root' was adopted for this purpose to reduce clutter in `/'.)

Directory	Description
/usr	Subdirectories with files related to *user* tools and applications.
/usr/X11R6	Files relating to the X Window System, including those programs (in `/usr/X11R6/bin') that run only under X.
/usr/bin	Tools and applications for users.
/usr/dict	*Dictionaries* and word lists (slowly being outmoded by `/usr/share/dict').
/usr/doc	Miscellaneous system *documentation*.
/usr/games	*Games* and amusements.
/usr/info	Files for the GNU *Info* hypertext system.
/usr/lib	*Libraries* used by tools in `/usr/bin'.

Directory	Description
/usr/local	*Local* files -- files unique to the individual system -- including local documentation (in `/usr/local/doc') and programs (in `/usr/local/bin').
/usr/man	The online *manuals*, which are read with the man command
/usr/share	Data for installed applications that is architecture-independent and can be *shared* between systems.
/usr/src	Program *source* code for software compiled on the system.
/usr/tmp	Another directory for *temporary* files.
/var	*Variable* data files, such as spool queues and log files.

Making an Empty File

- To create the file `a_fresh_start' in the current directory, type:

```
$ touch a_fresh_start [RET]
```

- To create the file `another_empty_file' in the `work/completed' subdirectory of the current directory, type:

```
$ touch work/completed/another_empty_file
  [RET]
```

Copy, Move or delete a File

- Copy the file passwd found in /etc

  $ cp /etc/passwd /home/mypasswd

- Move the file mypasswd into /etc with new name newpasswd

  $ mv /home/mypasswd /etc/newpasswd

- Rename
  - **$** mv /home/passwd /home/newpasswd
- Remove(delete) newpasswd

  $ rm /etc/newpasswd

- cp & mv & rm can be used for directories (the option –r for recursion)

Making a Directory

- To make a new directory called `work' in the current working directory, type:
  - **`$ mkdir work [RET]`**
- To make a new directory called `work' in the `/tmp' directory, type:
  - **`$ mkdir /tmp/work [RET]`**

Making a Directory Tree

- To make the `work/completed/2001' directory a subdirectory of the `completed' directory, which in turn is a subdirectory of the `work' directory in the current directory, type:

```
$ mkdir -p work/completed/2001 [RET]
```

Changing Directories

- To change the current working directory to `work', a subdirectory in the current directory, type:

```
$ cd work [RET]
```

- To change to the current directory's parent directory, type:

```
$ cd .. [RET]
```

- To change the current working directory to `/usr/doc', type:

```
$ cd /usr/doc [RET]
```

- To make your home directory the current working directory, type:

```
$ cd [RET]
```

- To return to the directory you were last in, type:

```
$ cd - [RET]
```

Getting the Name of the Current Directory

- To determine what the current working directory is, type:

```
$ pwd [RET]
/home/mrs
$
```

Listing Directories

- To list the contents of the current working directory, type:

```
$ ls [RET]
apple cherry orange
$
```

- To list the contents of `work', a subdirectory in the current directory, type:

```
$ ls work [RET]
```

- To list the contents of the `/usr/doc' directory, type:

```
$ ls /usr/doc [RET]
```

- To list the contents of the directory so that directories and executables are distinguished from other files, type:

```
$ ls -F [RET]
repeat* test1 test2 words/
$
```

Other options

- Listing Attributes: Listing file attributes.
  -l
- Listing Recursively: Listing directories and their subdirectories.
  -R (-r reversed order)
- Listing Newest: Listing newest files first.
  -t
- Listing Hidden: Listing hidden or special files.
  -a & -A (except '.' & '..')
- Listing Color: Directory listings in color.
  --color

Listing Directory Tree Graphs

- To output a tree graph of the current directory and all its subdirectories, type:

```
$ tree [RET]
.
|-- projects
|    |-- current
|    `-- old
|         |-- 1
|         `-- 2
`-- trip
     `-- schedule.txt
4 directories, 3 files
$
```

shell

Shell

- the program that reads your command input and runs the specified commands.
- The shell environment is the most fundamental way to interact with the system.
- You are said to be "in" a shell from the very moment you've successfully logged in to the system.
- Works a
  - Command interpreter
  - Programming environment (shell script)

Prompt

- The `$' character preceding the cursor is called the *shell prompt.*
- It tells you that the system is ready and waiting for input.
- On Debian systems, the default shell prompt also includes the name of the current directory .
- A tilde character (`~') denotes your home directory, which is where you'll find yourself when you log in.
- A number sign (`#') instead of a `$', means that you're logged in with the superuser, or root, account.

Environment Variables

- PATH : where to find commands
  - `$ echo $PATH`
- PS1 : shell prompt
  - `$ echo $PS1`
  - `$ PS1='hi'`
  - `$ PS1='$PWD>'`
- HOME : home directory
  - `$ echo $HOME`

Initialization File

- Executed every time the user logs on the system
- Path: ~/.bash_profile

Start Up file

- Executed every time the user starts a new shell
- Path: ~/.bashrc

Shell Examples

Shell	Initialization File	Startup File
sh	~/.profile	-----
ksh	~/.profile	~/.kshrc
csh	~/.login	~/.cshrc
bash	~/.bash_profile	~/.bashrc

Redirection

- Standard output
  - `$ Ls -R /etc > /home/logfile`
  - `$ Ls -R /etc >> /home/logfile`
- Error
  - `$ Ls -R /etc 2> /home/logfile`
  - `$ Ls -R /etc 2>> /home/logfile`
- Both
  - `$ Ls -R /etc &> /home/logfile`
  - `$ Ls -R /etc &>> /home/logfile`

User Administration

Create User Account

- **`$ useradd ali`**

- **`-c`** “comment”
- **`-g`** join him into group
- **`-d`** path of home directory
- **`-s`** default shell path
- **`-G`** groups to be member of

- **`$ groupadd students`**

/etc/passwd

7 fields:

- Username: 1-32 character
- X (removed password)
- User ID: UID 0 (zero) is reserved for root and UIDs 1-99 are reserved for other predefined accounts. Further UID 100-999 are reserved by system for administrative and system accounts/groups
- 'Primary Group' ID
- Comment
- Path of home directory
- Path of default shell

/etc/group

- *Group_name:* It is the name of group. If you run ls -l command, you will see this name printed in the group field.
- *Password:* Generally password is not used, hence it is empty/blank. It can store encrypted password. This is useful to implement privileged groups.
- *Group ID (GID):* Each user must be assigned a group ID. You can see this number in your /etc/passwd file.
- *Group List:* It is a list of user names of users who are members of the group. The user names, must be separated by commas.

/etc/shadow

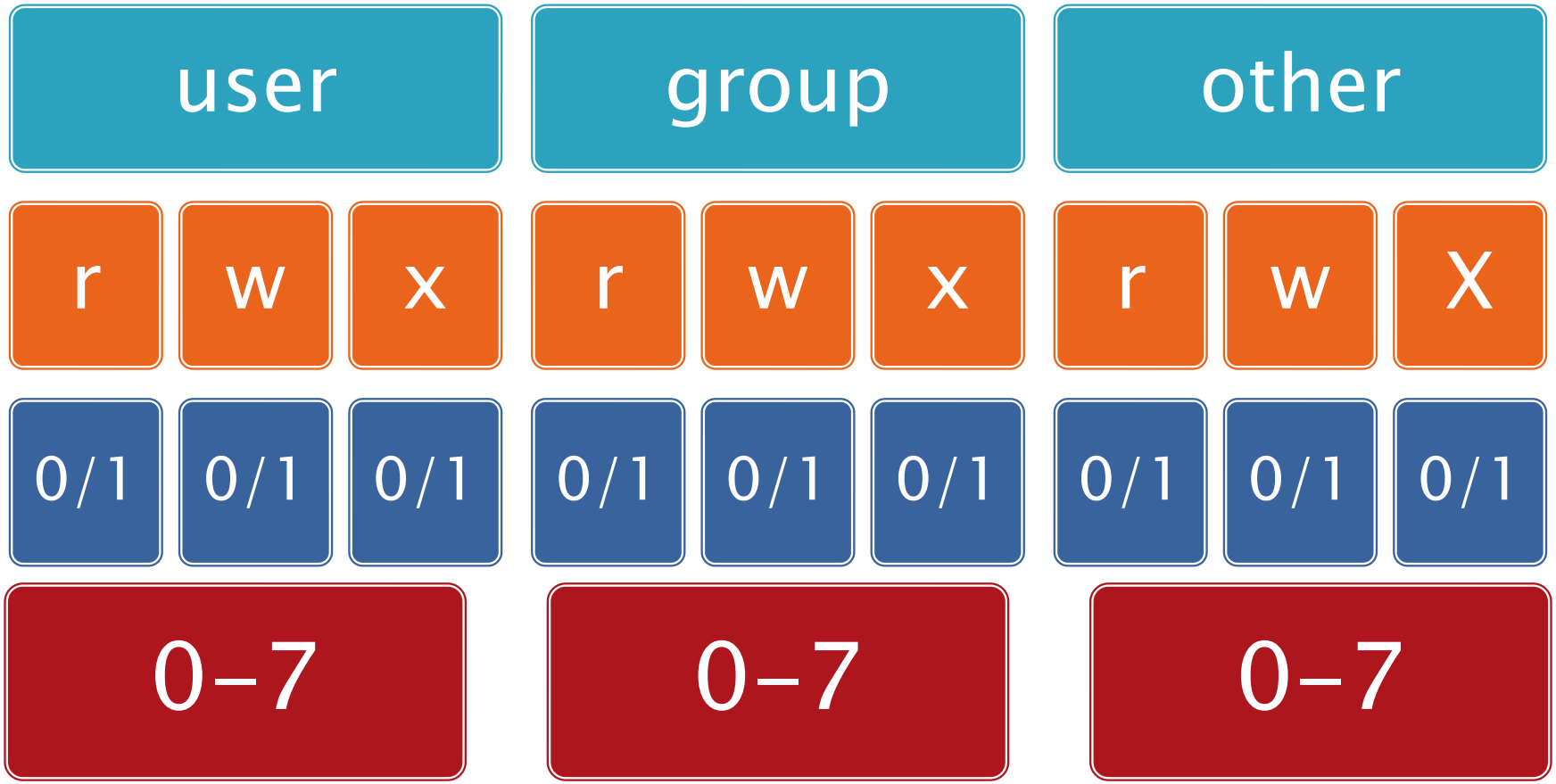
- *User name :* It is your login name
- *Password:* It your encrypted password.
- *Last password change* (lastchanged): Days since Jan 1, 1970 that password was last changed
- *Minimum:* The minimum number of days required between password changes
- *Maximum:* The maximum number of days the password is valid
- *Warn :* The number of days before password is to expire that user is warned
- *Inactive :* The number of days after password expires that account is disabled
- *Expire* : days since Jan 1, 1970 specifying when the login may no longer be used

File Security

Permissions

Permission	File	Directory
Read (r)	Display contents	List files
Write (w)	Modify	Add/remove files
Execute	Run	Enter (cd)

Permissions



File type

Symbol	Meaning
-	Regular file
d	Directory
l	Soft link
c or d	Device (SW driver)

Modify permissions

- `$ chmod u+x,g+x,o-r mypasswd`
- `$ chmod u=rwx,g=rx,o=r mypasswd`
- `$ chmod a+rwx mypasswd`
- `$ chmod 761 mypasswd`
- Make default permissions at creation

  rw- r-- ---

`$ umask 137`

Text Editing

Command line editor (vi)

- $ vi /home/myfile

- Modes:
  - Command: press esc
  - Typing: press i
- Save: esc :w
- Quit: esc :q
- Force: !

vi command mode

- a move cursor one place to the right and change into insert mode
- A move cursor to the end of the line and change into insert mode
- I move cursor to the start of the line and change into insert mode
- 0 move to the start of the line, no mode change
- $ move to the end of the line, no mode change

vi command mode

- w move forward from word to the next word
- b move backward from word to the prev. word
- yy copy line
- dd cut line
- p paste
- x delete till the end of the line then continue as backspace
- k move up
- j move down
- l move right
- h move left

GUI editor

- `$ gedit /home/myfile`

Linux File System

inode number

- A number uniquely identifies each file in the system.
  - `$ ls -i /etc/passwd`

links

- Hard link:
  - New name for the same file
  - 2 names & 1 inode number
  - **`$ ln ff hlff`**
- Soft link:
  - New file acts as a pointer points to another file
  - 2 files & 2 different inode numbers
  - **`$ ln -s ff slff`**

Process Management

Process information

- Full information about my processes
  - **`$ ps -f`**
- All processes
  - **`$ ps -e`**
- Specific process
  - **`$ ps -p`** ***processid***

Signals

Signal No.	Signal Name	Signal Effect
2	INTERRUPT	Pause process, can be ignored
9	KILL	End process
15	TERMINATE	End process, can be ignored
19	STOP	Pause process
18	CONTINUE	Start stopped process

Send Signals to process

- `$ kill -STOP  315`
- `$ kill -19    315`

- `$ pkill -STOP vi`
- `$ pkill -19   vi`

- `$ ps -e | grep vi`
- `$ pgrep vi`

Data compression

Compression

gzip	bzip2	zip
Lossless	Lossless	Lossless
Delete source	Delete source	Keep source
Multi-multi	Multi-multi	Multi-single
-----.gz	------.bz2	------.zip
gunzip	bunzip2	unzip

Archiving

- Put a directory in the form of a file with no compression(optional) to backup on tapes
- -z for gzip & -j for bzip2

- **$ tar -cf mybackup.tar myfile**
- **$ gzip mybacup.tar**

- **$ gunzip mybackup.tar.gz**
- **$tar -xf mybackup.tar**

http://www.cyberciti.biz/
http://dsl.org/cookbook/cookbook_toc.html